Escuela de Arquitectura Tecnológica

Modernización evolutiva de sistemas legados componibles

Este e-book explora cómo llevar sistemas legados hacia arquitecturas modernas de manera gradual y estratégica, abordando patrones de modernización y técnicas de integración para transformar plataformas sin interrumpir la operación del negocio.

Hacemos uso de:

Introducción a la modernización evolutiva desistemas legados componibles

La modernización evolutiva de sistemas legados componibles es una estrategia para modernizar la tecnología de tu organización forma gradual y modular. En lugar de realizar una transformación total de una sola vez, esta metodología enfatiza la evolución del sistema en partes manejables, adaptando módulos o componentes independientes que puedan integrarse progresivamente con nuevas arquitecturas. En particular, "componibles" sugiere que el enfoque busca diseñar o actualizar los sistemas de forma que los diferentes módulos o funcionalidades sean autónomos y puedan combinarse o reemplazarse sin afectar el sistema entero, facilitando la flexibilidad y escalabilidad a lo largo deltiempo.

¿Qué son sistemas legados

Los sistemas legados se refieren a aplicaciones, plataformas o infraestructurastecnológicas antiguas que todavía son críticas para el funcionamiento de una organización, pero que presentan limitaciones significativas en términos deadaptabilidad, escalabilidad, y mantenimiento. Estos sistemas suelen haberseconstruido hace muchos años, con tecnologías, arquitecturas y prácticas que hoyestán obsoletas o que ya no son compatibles con las necesidades modernas delnegocio. Se caracterizan por:

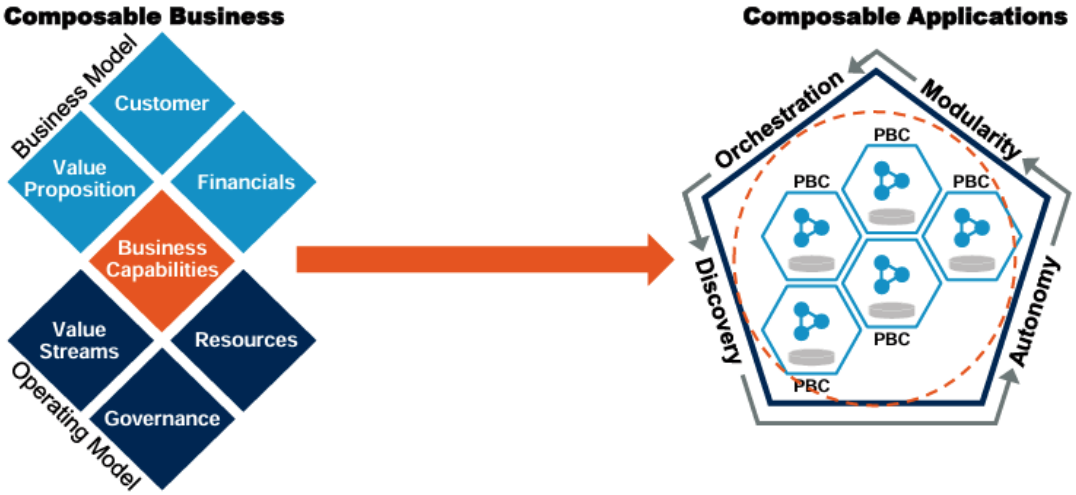
1. **Tecnología obsoleta:** usan lenguajes de programación, frameworks, o bases de datos antiguos que dificultan la incorporación de nuevos talentos y la integración con tecnologías modernas.
2. **Rigidez y falta de modularidad:** a menudo, son monolíticos o están fuertemente acoplados, lo cual impide modificar partes del sistema sin afectar otras.
3. **Altos costos de mantenimiento:** debido a la obsolescencia y la falta de expertos en la tecnología, mantener el sistema puede resultar costoso y arriesgado.
4. **Dependencia organizacional:** aunque presentan limitaciones, suelen ser críticospara los procesos operativos de la organización, lo que dificulta su reemplazo completo.

¿Qué son sistemas componibles?

Son un término acuñado por gartner (composable business), se refieren a unaarquitectura de ti que permite que las aplicaciones y sistemas se construyan,reconfiguren y actualicen a partir de módulos o componentes independientes quepueden ensamblarse y adaptarse de manera flexible. Esta arquitectura promuevela agilidad, escalabilidad y adaptabilidad al facilitar que los sistemas secompongan de elementos intercambiables, sin estar fuertemente acoplados.

Gartner describe los sistemas componibles basados en tres principios fundamentales que los hacen particularmente efectivos en entornos demodernización evolutiva:

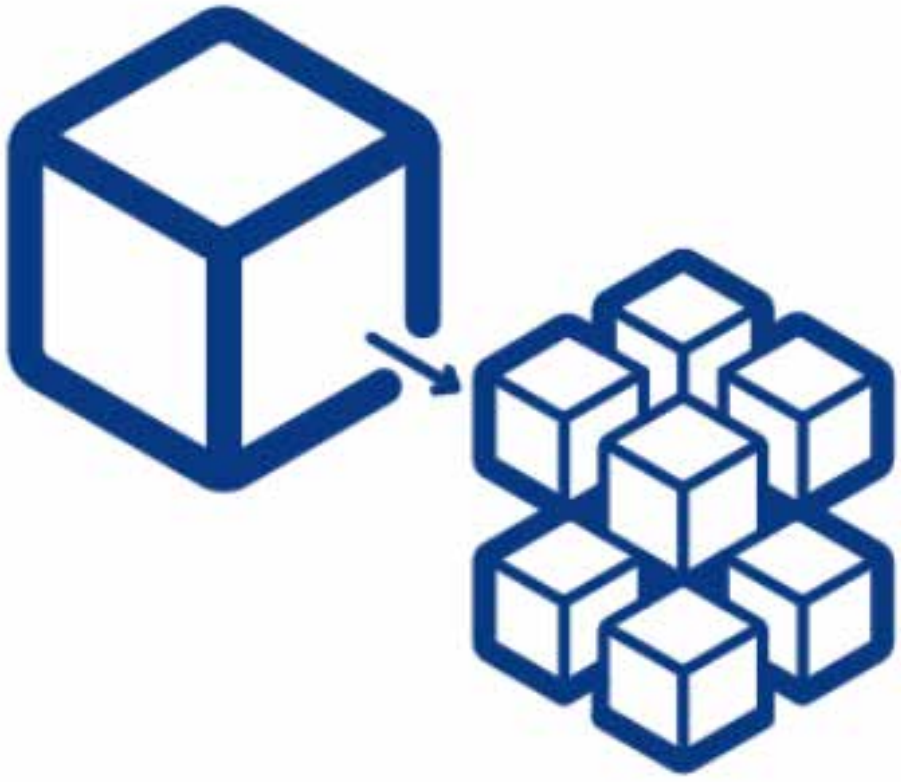
1. **Modularidad:** los sistemas componibles están diseñados a partir decomponentes independientes que cumplen funciones específicas. Esta modularidad facilita la personalización, el mantenimiento y la sustitución departes sin afectar el sistema completo.
2. **Orquestación:** los componentes se integran de forma flexible y pueden interactuar a través de interfaces y apis estándar. Esto permite la coordinación y flujo de trabajo entre módulos, manteniendo la independencia y cohesión de cadauno.
3. **Descubrimiento:** los componentes son fácilmente localizables y reutilizables, loque acelera el desarrollo y la innovación. Los equipos pueden acceder a módulos existentes y recombinarlos en nuevas configuraciones para responder rápidamente a los cambios en el negocio.



¿Por qué son relevantes los sistemas componibles en la modernización de ti?

En el contexto de la modernización de sistemas legados, la componibilidad permite descomponer un sistema monolítico o legado en módulos funcionales individuales. Esto facilita el proceso de actualización, permitiendo integrar componentes modernos en lugar de reemplazar el sistema entero de una vez. La arquitectura componible apoya la evolución gradual del sistema, lo cual minimiza riesgos, reduce costos y asegura una continuidad operativa durante el proceso de modernización.

Los sistemas componibles son, en esencia, el camino hacia una arquitectura ágil y escalable, donde las organizaciones pueden experimentar, innovar y adaptarse alos cambios sin estar limitadas por las barreras de los sistemas monolíticos tradicionales.



¿Por qué modernizar?

La modernización de sistemas legados es crucial para las organizaciones quebuscan mantenerse competitivas, optimizar sus operaciones y responder deforma ágil a las demandas del mercado.

1. **Reducción de riesgos operacionales y de seguridad:** los sistemas legados, al estar construidos con tecnologías antiguas, suelen tenervulnerabilidades de seguridad que ya no son compatibles con los estándaresactuales.
2. **Agilidad y escalabilidad:** las arquitecturas antiguas son menos flexibles, dificultando la adaptación acambios o la implementación de nuevas funcionalidades.
3. **Reducción de costos:** los sistemas legados suelen implicar altos costos de mantenimiento, tanto por ladificultad de encontrar personal capacitado en tecnologías obsoletas.
4. **Mejora de la experiencia del cliente y competitividad:** los sistemas antiguos pueden limitar la capacidad de las empresas para ofreceruna experiencia de usuario moderna y rápida.
5. **Preparación para la innovación:** los sistemas modernos facilitan la incorporación de tecnologías avanzadas, comointeligencia artificial, big data y automatización, que son esenciales para latransformación digital.
6. **Cumplimiento regulatorio y normativo:** con el tiempo, los estándares de cumplimiento y regulación cambian y, amenudo, los sistemas antiguos ya no cumplen con estas normativas.
7. **Continuidad y disponibilidad del negocio:** los sistemas obsoletos son propensos a fallos y tiempos de inactividad, lo cualpuede impactar la continuidad del negocio.

Modernización evolutiva v Transformaciones big bang

A la hora de modernizar puedes elegir entre modernizar de manera evolutiva o hacer una transformación big bang, la diferencia radica en el enfoque y ritmo con el cual se modernizan sistemas legados o aplicaciones antiguas.

Modernización evolutiva

Este enfoque consiste en realizar cambios graduales y continuos en el sistema, actualizando partes individuales o módulos de forma secuencial permitiendo que el sistema legado se transforme progresivamente, mientras coexisten componentes modernos y antiguos. Los cambios se introducen con menos riesgo y menor interrupción al negocio, ya que el sistema continúa funcionando durante el proceso.

Ventajas:

- Minimiza riesgos, ya que permite pruebas y correcciones sin afectar el sistema completo.
- Adapta gradualmente el sistema a nuevas tecnologías y regulaciones.

Desventajas:

- Puede ser un proceso más largo que requiere una visión y un compromiso a largo plazo.

Transformación big bang

En una transformación "big bang", el sistema legado es reemplazado completamente de una sola vez, trasladando todas las funciones y operaciones alnuevo sistema en una implementación única. Este enfoque generalmente implicauna desconexión total del sistema antiguo y el inicio de una nueva arquitectura.

Ventajas:

- Permite la adopción rápida de tecnologías modernas y nuevas arquitecturas. Una vez completado, no necesita operar y mantener un sistema legado.

Desventajas:

- Riesgo alto de interrupción del negocio en caso de problemas o fallas.
- Requiere una inversión significativa de recursos en un solo momento.

Estrategias para modernizar sistemas legados

Existen diversas estrategias que pueden emplearse, cada una con sus enfoques y patrones específicos, que permiten adaptar, migrar y mejorar la arquitectura desistemas existentes.

1. Replatform

El replataform es un enfoque que implica migrar una aplicación a una nueva plataforma tecnológica sin cambiar el código base. Esto puede incluir mover aplicaciones de un entorno local a la nube, lo que ofrece beneficios como escalabilidad, resiliencia y acceso a servicios modernos, pero no mejora la flexibilidad, mantenibilidad y adaptabilidad del sistema

2. Service wrapping

Esta estrategia consiste en envolver las funciones de un sistema legado en servicios web. Permite que las aplicaciones modernas interactúen con el sistema antiguo a través de apis, sin necesidad de modificar el código legado. Esto facilitala integración de nuevas aplicaciones sin alterar la funcionalidad del sistema existente, sin embargo no mejora la disponibilidad o escalabilidad del sistema.

3. Refactorización

La refactorización implica mejorar el código existente modificándolo para mejorar su estructura y mantenibilidad sin alterar su funcionalidad externa. Este enfoquees ideal cuando se desea optimizar el rendimiento o simplificar el código sin perder las capacidades existentes, este enfoque mejora la mantenibilidad perohace el sistema adaptable y flexible

4. Reescritura

La reescritura implica reconstruir la aplicación desde cero, lo que puede sernecesario en casos donde la deuda técnica es tan alta que los costos demantenimiento superan los beneficios. La reescritura se puede dar bajo 2 modalidades:

- **Strangler application:** este patrón propone dividir el sistema monolítico en componentes más pequeños y manejables de manera gradual.
- **Rip and replace:** implica desechar completamente el sistema legado y reemplazarlo con una nueva solución.

Ventajas y desventajas de cada Estrategia deevolución

Estrategia	Ventajas	Desventajas
Replatform	- Migración a nueva plataforma sin cambiar código base.	- No mejora la flexibilidad, mantenibilidad ni adaptabilidad del sistema.
	- Acceso a servicios modernos y escalabilidad.	- Dependencia de la nueva plataforma, lo que puede generar desafíos adicionales.
	- Mejora la resiliencia al utilizar infraestructura moderna.	
Service Wrapping	- Permite la integración de nuevas aplicaciones sin modificar el código legado.	- No mejora la disponibilidad ni escalabilidad del sistema existente.
	- Facilita la comunicación entre sistemas a través de APIs.	- Puede introducir complejidades en la gestión de servicios y dependencias.
	- Conserva la funcionalidad del sistema legado.	
Refactorización	- Mejora la estructura y mantenibilidad del código existente.	- No transforma radicalmente la arquitectura del sistema, limitando su adaptabilidad.
	- Optimiza el rendimiento sin alterar la funcionalidad externa.	- Puede requerir tiempo significativo y recursos para realizar adecuadamente.
	- Facilita futuras modificaciones y mejoras.	
Reescritura	- Permite la reconstrucción de la aplicación desde cero, eliminando deuda técnica.	- Costoso y arriesgado, puede provocar interrupciones en el servicio.
	- Se puede dar bajo diferentes modalidades.	- Requiere una planificación y capacitación extensa para los usuarios.
	- Puede mejorar significativamente la flexibilidad y adaptabilidad del sistema.	- Puede llevar tiempo y recursos significativos.
Strangler Application	- Permite una migración gradual, minimizando riesgos.	- La complejidad de gestionar un sistema híbrido (nuevo y legado) puede ser alta.
	- Facilita la transición sin interrumpir el servicio.	- Puede requerir tiempo prolongado para completar la modernización.
Rip and Replace	- Ofrece una solución limpia y actualizada sin limitaciones del sistema legado.	- Implica un alto costo inicial y riesgo de pérdida de datos o funcionalidades.
	- Permite adoptar tecnologías y arquitecturas modernas.	- Puede generar resistencia al cambio por parte de los usuarios.

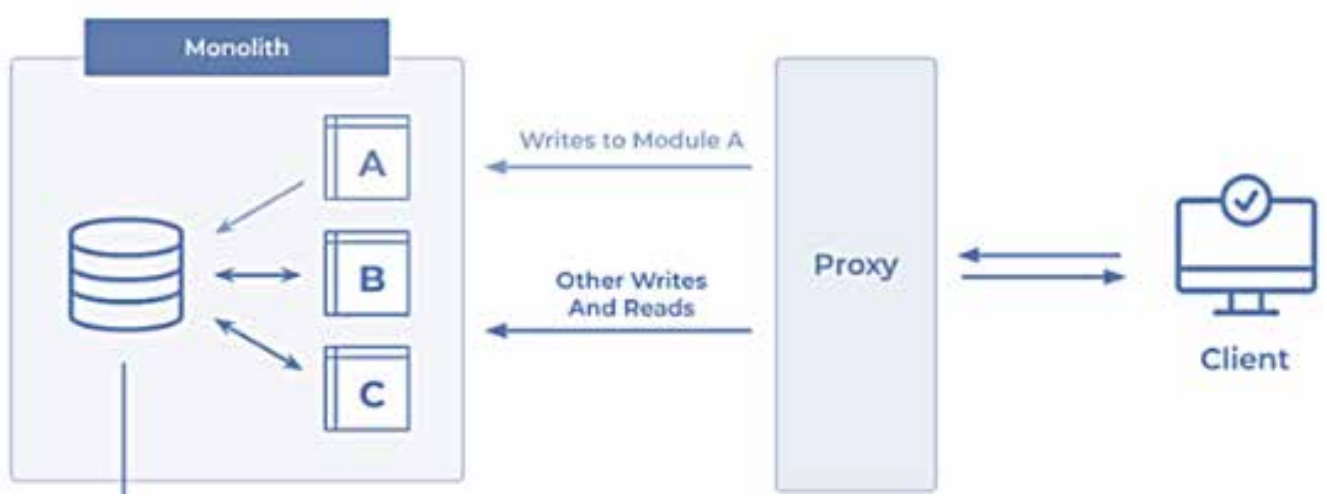
Patrón strangler application

El patrón strangler application es una estrategia de modernización de softwareque permite migrar de un sistema monolítico a una arquitectura más moderna yflexible, como microservicios, de manera gradual y sin necesidad de detener laoperación del sistema existente.

El patrón strangler application sugiere ejecutar 4 etapas para migrarfuncionalidades.

Etapa 1 - identificación de componentes

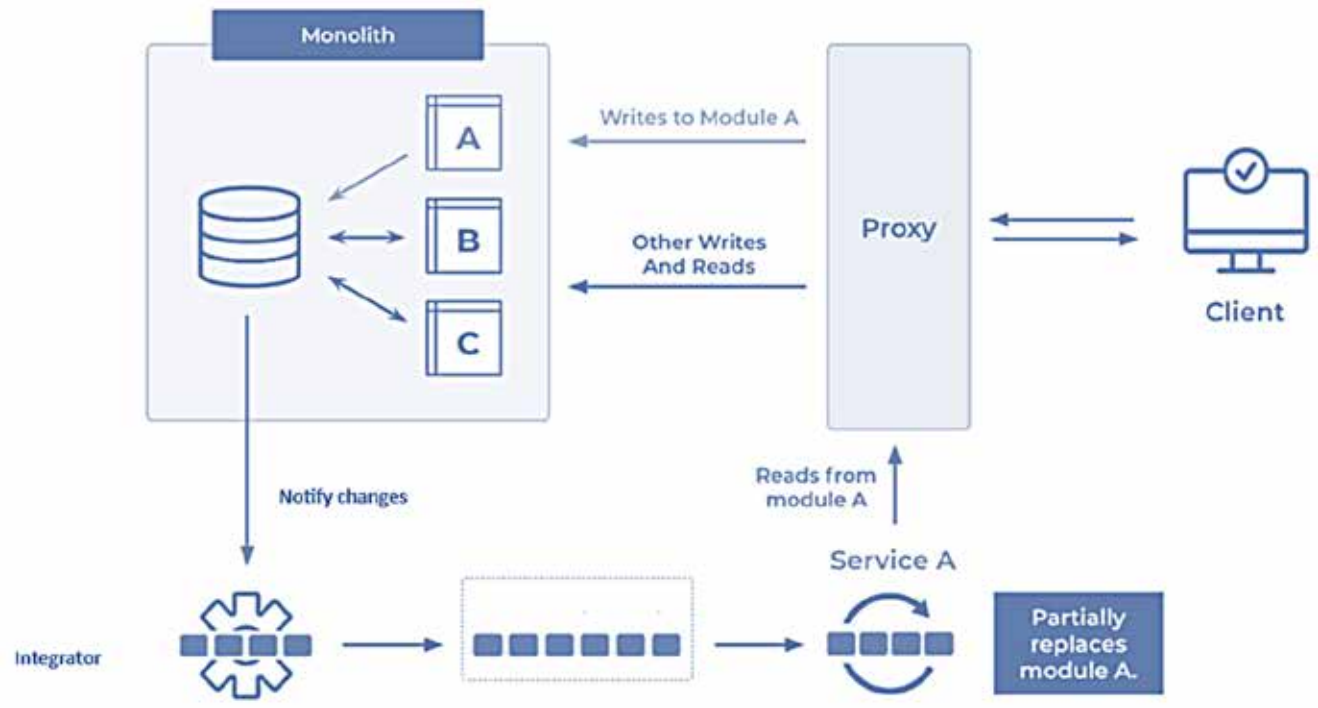
Se analizan las funcionalidades delsistema monolítico y se identifican aquellas que pueden ser aisladas yreemplazadas. En la siguiente imagen se puede ser un sistema legado compuestopor 3 módulos a, b y c que reciben tráfico de aplicaciones cliente ya sean frontends, apps u otros servicios.



Etapa 2 - creación de nuevos servicios coexistiendo con el legado

Se desarrollannuevos microservicios que replican la funcionalidad del legado. En este punto laaplicación nueva existe al mismo tiempo que existe la funcionalidad en el legado yel golden record de la información aún es el sistema legado. Por esta razón esnecesario construir sincronizadores para que los datos insertados en el sistemalegado sean replicados al nuevo servicio.

En la siguiente imagen se puede ver que se implementa la lógica del módulo a enun nuevo servicio llamado service a, y para que esto pueda ser posible, el sistemalegado notifica los cambios en sus datos para que el servicio a pueda operaradecuadamente mientras las aplicaciones clientes usan el servicio a para consultar.



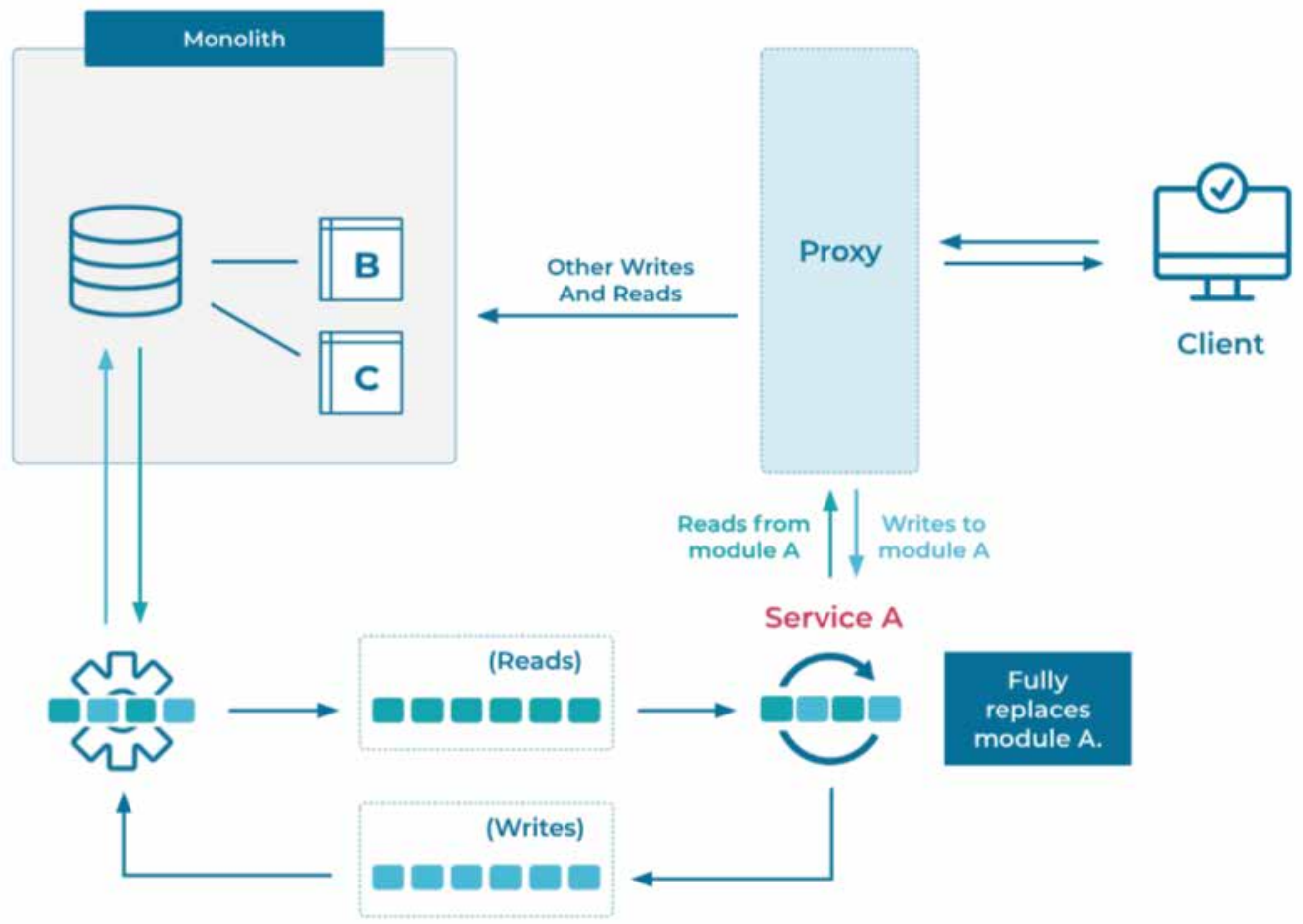
Qué es el golden record de los datos?

Un golden record (o "registro dorado") es un concepto en la gestión dedatos que se refiere a la versión más confiable y completa de un conjuntode datos sobre un objeto o entidad. Se utiliza para asegurar que todas lasáreas de una organización trabajen con la misma información precisa y actualizada.

Etapa 3 - redirección del tráfico al nuevo servicio

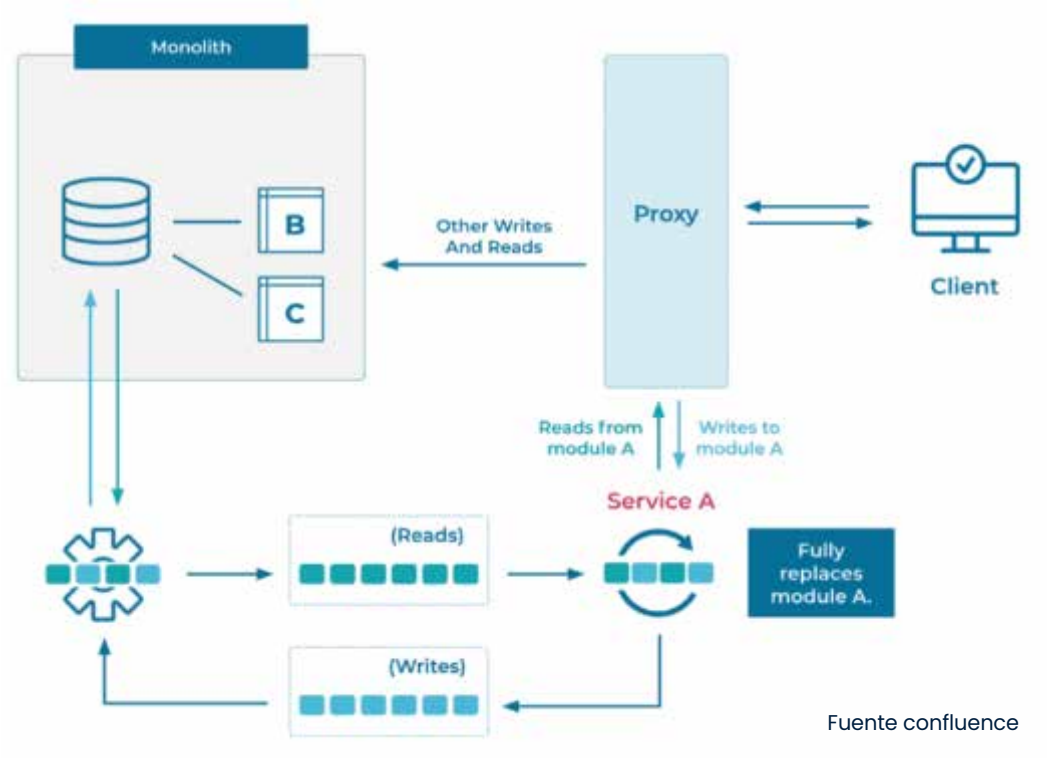
A medida que se desarrollan yprueban los nuevos microservicios, se redirige el tráfico del sistema antiguo a losnuevos servicios de forma controlada. En esta fase el golden record de los datoses el nuevo servicio y el legado contiene solo en una vista materializada de ellospara que los demás procesos que se encuentran en él funcionen.

En la siguiente imagen se puede ver que el proxy redirige el tráfico al nuevosistema (service a) tanto para lectura como para escritura, en este punto elcódigo fuente que implementa el módulo a se mantiene como backup pero no esoperativo y aparece un mecanismo de sincronización de datos desde el nuevoservicio hacia el legado, para insertar en las antiguas tablas del modelo de datosy mantener funcionales los módulos b y c en los puntos donde los requieren



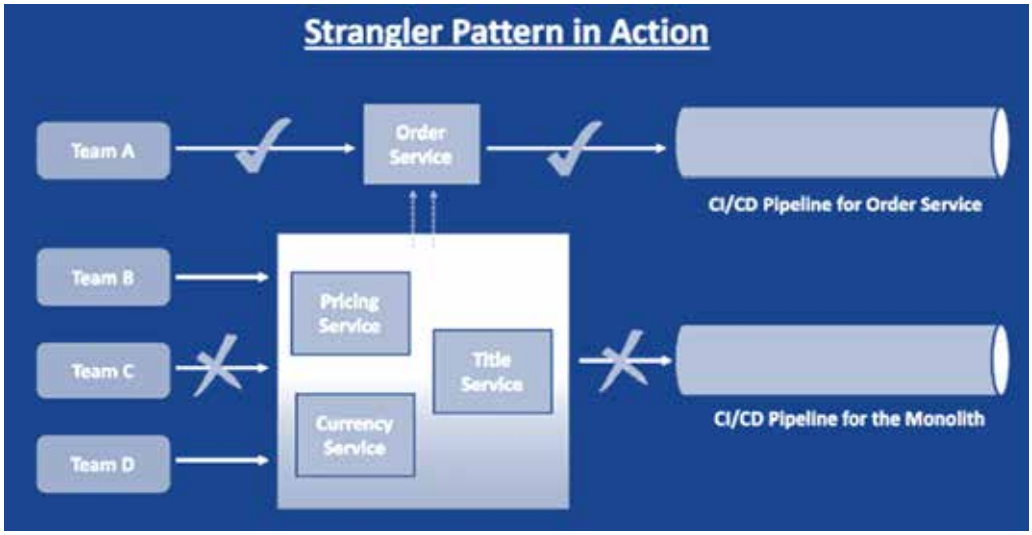
Etapa 4 - eliminación del código en el sistema legado

Una vez estabilizada la soluciónse elimina el código del sistema monolítico y se dejan los integradores mientrasotros módulos del sistema legado requieran los datos.



Fuente confluence

El ciclo continúa hasta que todos los módulos sean migrados y el monolito demanera natural deje de existir. Cómo se puede ver en la imagen, en un primermomento todos los módulos se encuentra juntos y al extraerse y pasar por los 4 pasos, empiezan a ser independientes.



Uso de why driven design para diseñar la arquitectura de migración de sistemas legados de manera evolutiva

Why driven design es conjunto de herramientas, técnicas metodológicas yestrategias para hacer diseño de arquitectura de soluciones ágil. Whydd puedeser adoptado como marco de diseño de arquitectura evolutiva, para más detalles puedes ver el *e-book why driven design para diseño de arquitectura de soluciones ágil.*

Estrategias de sincronización de datos

En la evolución de sistemas legados, la sincronización de datos es uno de los desafíos técnicos y operativos más críticos. A medida que se modernizan componentes de un sistema o se migran hacia arquitecturas distribuidas, es esencial garantizar que los datos se mantengan consistentes, precisos y disponibles en todo momento. Este capítulo explora cómo la sincronización dedatos se convierte en un eje central para la evolución exitosa de sistemas, permitiendo la coexistencia de componentes antiguos y nuevos sin pérdida deintegridad ni de funcionalidad.

Desafíos en la sincronización de datos en sistemas legados

1. **Resolución de conflictos:** cómo manejar inconsistencias entre datos sincronizados y desincronizados; reglas y políticas de resolución de conflictos.
2. **Manejo de latencia y consistencia de datos:** impacto de la latencia en la integridad de los datos; soluciones de compensación.
3. **Compatibilidad entre modelos de datos antiguos y nuevos:** cómo sincronizar datos entre diferentes versiones de bases de datos y estructuras de datos.
4. **Problemas de seguridad y gobernanza de los datos:** asegurar la integridad y privacidad de los datos durante la sincronización.

Modelos de sincronización de datos

Los dos modelos más comunes para esta sincronización son la sincronización en tiempo real y la sincronización programada (batch processing), ambos con características particulares que los hacen más o menos adecuados para ciertostipos de sistemas y operaciones.

1. Sincronización en tiempo real: la sincronización en tiempo real implica que los datos se transmiten y actualizan instantáneamente entre sistemas ocomponentes en el momento en que se produce un cambio. Esta modalidad escomún en sistemas que requieren una interacción inmediata, como aplicaciones bancarias, plataformas de e-commerce o sistemas de monitoreo en salud.

Ventajas de la sincronización en tiempo real

1. **Actualización inmediata:** permite que los sistemas tengan acceso a datosprecisos y actualizados en todo momento, ideal para aplicaciones que dependen de decisiones basadas en datos en tiempo real.
2. **Mejor experiencia de usuario:** al ofrecer datos actualizados instantáneamente, los usuarios tienen una interacción continua y sin interrupciones, especialmente beneficioso en sistemas críticos.
3. **Reducción de duplicidad:** al procesar la información conforme se genera, disminuye la posibilidad de acumulación de datos repetidos o inconsistentes entre los sistemas.

Desventajas de la sincronización en tiempo real

1. **Requerimientos de infraestructura:** generalmente, este tipo de sincronización exige una infraestructura robusta que soporte un flujo constante de datos.
2. **Complejidad de integración:** la integración de sistemas en tiempo real puede requerir una alta inversión en programación y recursos para garantizar la consistencia y velocidad en todas las aplicaciones.
3. **Mayor carga en el sistema:** la transmisión continua de datos puede impactar el rendimiento del sistema, especialmente si hay grandes volúmenes de datos o múltiples puntos de integración.

Ejemplos de uso ideal

1. Plataformas bancarias donde las transacciones necesitan reflejarse inmediatamente en la cuenta del usuario.
2. Aplicaciones de e-commerce donde el inventario debe actualizarse en tiempo real para reflejar las compras y la disponibilidad.
3. Sistemas de salud en tiempo real donde los datos clínicos deben sincronizarse de forma continua para asistir en decisiones críticas de los médicos.
4. Sistemas iot donde se reciben datos se sensores y dispositivos de campo para tomar decisiones en tiempo real.

2. Sincronización programada (batch processing): la sincronización programada, también conocida como procesamiento por lotes, implica que los datos se sincronizan en intervalos regulares, como cada hora, cada día o cada semana. En lugar de actualizarse en el momento exacto de un cambio, se acumulan en unlote y se procesan todos juntos. Este método es ideal para sistemas donde los datos no necesitan estar actualizados inmediatamente o donde la sincronizacióncontinua no es necesaria.

Ventajas de la sincronización programada

- **Reducción de carga en el sistema:** procesar los datos en lotes permite controlar la carga en el sistema, ya que las actualizaciones son periódicas y no constantes.
- **Menor costo de infraestructura:** los sistemas de sincronización programada suelen necesitar menos recursos, ya que no requieren infraestructura de baja latencia o alta disponibilidad para procesar los datos de forma continua.
- **Flexibilidad en el manejo de datos:** al trabajar en lotes, se pueden realizar verificaciones de integridad y optimizaciones durante la sincronización que no serían posibles en tiempo real.

Desventajas de la sincronización programada

- **Retraso en la actualización de datos:** la naturaleza de este método implicaque los datos no están actualizados en tiempo real, lo cual puede ser una desventaja si se requiere información inmediata.
- **Complejidad de programación:** si los sistemas son altamente dependientes entre sí, coordinar el procesamiento de lotes puede ser complicado y requiereuna lógica cuidadosa.
- **Mayor riesgo de conflictos de datos:** en entornos donde los datos son actualizados por múltiples sistemas, los cambios conflictivos pueden ser más difíciles de resolver en sincronizaciones programadas.

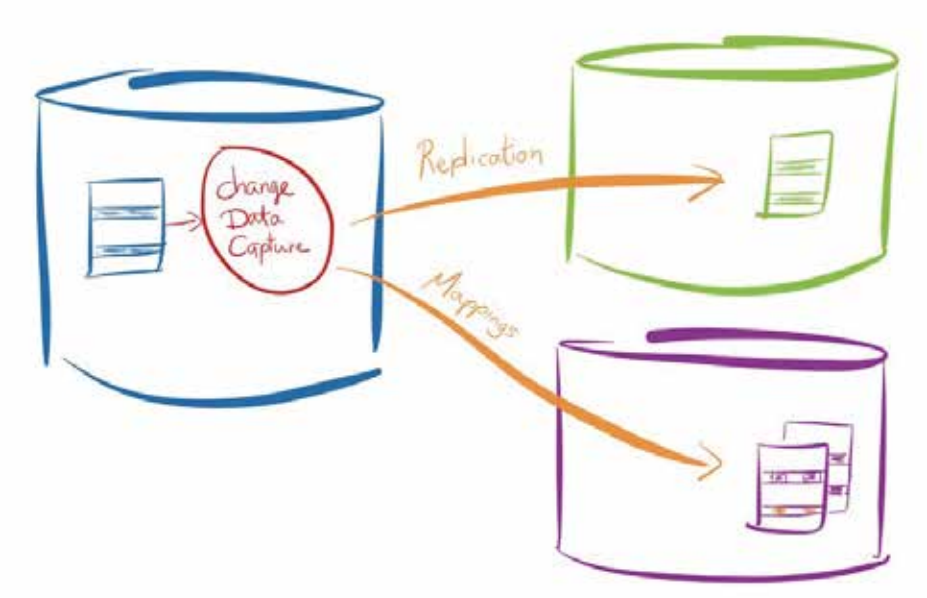
Ejemplos de uso ideal

- **Sistemas de reporting** donde los informes diarios o semanales son suficientes para tomar decisiones.
- **Procesos de payroll** donde la información sobre horas trabajadas y pagos se sincroniza mensualmente.
- **Aplicaciones de analítica de datos** donde los datos recolectados pueden acumularse y procesarse en lotes sin requerir actualizaciones en tiempo real.

Herramientas y arquitecturas desincronización de datos

Change data capture para sincronizar datos en tiempo real:

Change Data Capture (CDC) es una técnica de integración de datos que permite capturar y transmitir cambios en tiempo real desde una base de datos origen hacia un destino, facilitando la sincronización continua de datos entre sistemas. Se basa en leer los logs de transacciones de la base de datos (si es relacional) o aplicar técnicas de rastreo en bases de datos NoSQL para identificar y capturar cambios. Los eventos de cambio se pueden transmitir a un sistema de destino, como un servicio en la nube, una base de datos de nueva generación o una aplicación moderna, logrando que el flujo de datos entre sistemas legados y nuevas aplicaciones se mantenga sincronizado.



Con CDC, las organizaciones pueden:

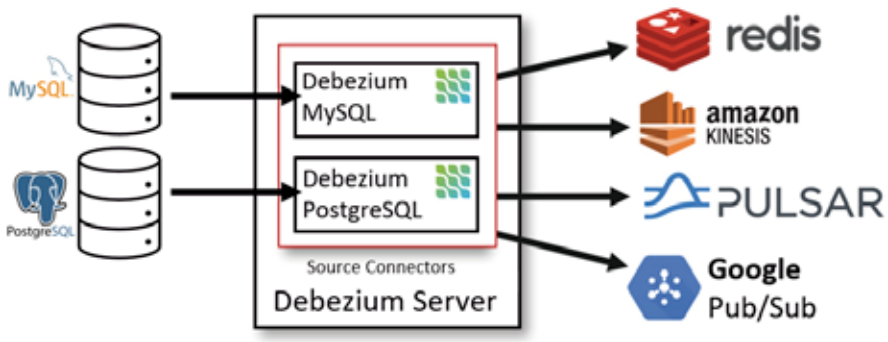
- **Asegurar la integridad de los datos:** A medida que se realiza la migración de funcionalidades del sistema legado al nuevo sistema, CDC garantiza que cualquier cambio en los datos se refleje en ambos sistemas.
- **Minimizar el impacto en la base de datos:** CDC captura solo los cambios necesarios, lo que reduce la carga en la base de datos comparado con procesos de replicación completa o de actualización masiva.
- **Facilitar la flexibilidad de la arquitectura:** Con CDC, los equipos de TI pueden migrar partes del sistema a una nueva aplicación sin interrumpir la operación de otros módulos, permitiendo una modernización gradual.

Debezium para implementar change data capture:

Debezium es un conjunto de servicios distribuidos que capturan los cambios anivel de fila en las bases de datos y los envía a sistemas externos con unavelocidad Near real time.

Debezium facilita la implementación de CDC mediante:

- **Lectura de logs de transacción:** Utiliza los logs de transacción de la base dedatos para capturar cambios sin impacto en el rendimiento.
- **Publicación de eventos en tiempo real:** Los cambios se envían como eventosfacilitando que otros sistemas los consuman en tiempo real.
- **Configuración y monitoreo simplificados**: Debezium permite configurar lacaptura de cambios de forma rápida, además de monitorear el flujo deeventos para garantizar que no existan inconsistencias de datos.



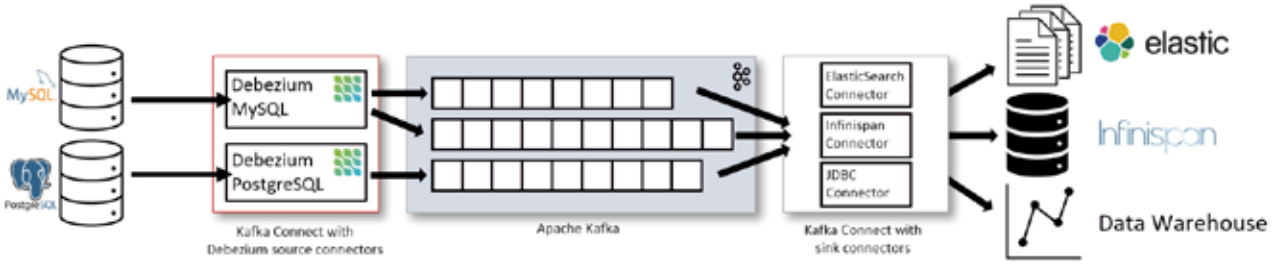
Consideremos una institución financiera que desea migrar sus operaciones de unsistema legado de gestión de cuentas bancarias a una nueva aplicaciónmoderna. CDC permite:

- **Mantener las cuentas sincronizadas en tiempo real:** Cada cambio en losdatos de cuentas bancarias en el sistema legado se captura y refleja en lanueva aplicación.
- **Permitir que nuevas funcionalidades operen en el nuevo sistema:** Con CDC,funcionalidades como la consulta de saldos o el historial de transaccionespueden comenzar a ejecutarse en la nueva aplicación mientras los datos semantienen actualizados desde el sistema antiguo.
- **Migrar gradualmente los procesos de forma controlada:** Con la sincronizaciónde datos en tiempo real, el equipo de TI puede elegir qué módulos moverprimero y cuándo, sin impactar negativamente al usuario final ni afectar laintegridad de los datos.

Kafka conkect para implementar change data capture:

Kafka Connect es una herramienta nativa de Apache Kafka diseñada para simplificar la integración de datos entre Kafka y otros sistemas. Funciona mediante conectores preconstruidos o personalizados que se encargan de extraer y cargar datos desde y hacia Kafka, permitiendo la creación de pipelines de datos en tiempo real.

En el contexto de Change Data Capture, Kafka Connect monitorea y captura eventos de cambios en las bases de datos, transmitiéndolos en tiempo real hacia otros sistemas a través de tópicos de Kafka.



Imaginemos una empresa bancaria que desea migrar su sistema de gestión de cuentas de un sistema monolítico legado a una aplicación moderna basada en microservicios. Para garantizar una transición sin problemas, se utiliza Kafka Connect con CDC para capturar cualquier cambio en la base de datos del sistema legado y replicarlo en la nueva base de datos de la aplicación moderna.

1. **Conexión con el sistema legado:** Kafka Connect se conecta a la base de datos del sistema legado mediante un conector CDC que captura cambios en tablas relevantes (por ejemplo, transacciones y datos de clientes).
2. **Transmisión a través de Kafka:** Cada cambio detectado se transmite como un mensaje en un tópico de Kafka. Estos mensajes se estructuran de modo que reflejen el evento exacto (inserción, actualización, eliminación) y la información afectada.
3. **Consumo en la nueva aplicación:** La nueva aplicación financiera consume estos mensajes desde Kafka, actualizando su base de datos y manteniendo sincronizada la información en tiempo real.
4. **Transición de funcionalidades:** A medida que la nueva aplicación se estabiliza, más funcionalidades se derivan hacia ella, mientras Kafka Connect continúa sincronizando los datos para mantener la consistencia entre ambos sistemas.

ETL low code para procesamiento en batch

El proceso de ETL (Extracción, Transformación y Carga) es una de las opciones más populares para sincronizar datos entre sistemas, especialmente cuando se modernizan aplicaciones antiguas o legadas hacia soluciones más modernas queno requieren procesamiento en tiempo real. Con herramientas
low code de ETL, como Apache NiFi, las organizaciones pueden simplificar el proceso de integraciónde datos y realizar una transición gradual de sistemas antiguos a nuevos, lo cual esideal en una arquitectura evolutiva y en la implementación del patrón StranglerApplication.

Las herramientas de ETL low code son plataformas que permiten diseñar y gestionar flujos de datos sin la necesidad de codificación extensa. En lugar de escribir código para cada paso del ETL, los usuarios configuran visualmente los procesos mediante arrastrar y soltar, seleccionando opciones desde una interfaz gráfica. Estas herramientas permiten:

- **Automatizar la integración de datos** desde diversas fuentes, incluyendo bases de datos, APIs, archivos y servicios en la nube.
- **Transformar los datos en tiempo real** para su estandarización, enriquecimiento y optimización antes de enviarlos al sistema de destino.
- **Orquestar flujos de datos complejos** sin necesidad de programación intensiva, facilitando la integración y sincronización de sistemas legados con nuevos sistemas.

Imaginemos una empresa que necesita migrar su sistema de gestión de clientes de una aplicación monolítica legada hacia un nuevo CRM basado en la nube. La herramienta de ETL low code permite:

1. **Extraer información de clientes** (nombres, contactos, historial de compras) desde el sistema legado sin modificarlo.
2. **Transformar estos datos** para adaptarse al formato del nuevo CRM (por ejemplo, adecuando el formato de fechas o unificando registros duplicados).
3. **Cargar los datos en el nuevo CRM** y actualizar el sistema legado de manera bidireccional hasta que la migración se haya completado totalmente.

Caso de estudio: extracción del módulo de gestión de cuentas desde un sistema monolítico

Cangre corp, es una entidad financiera con más de 30 años en el mercado, ofreciendo una amplia gama de productos y servicios bancarios, incluyendo cuentas de ahorro, cuentas corrientes, préstamos personales, tarjetas de crédito, y servicios de inversión. La empresa ha experimentado un crecimiento continuo, pero su sistema core bancario, desarrollado hace más de 15 años, se ha vuelto un obstáculo para la agilidad y la innovación debido a su arquitectura monolítica. Por esto han decidido modernizarlo, comenzando con la extracción del módulo de Gestión de Cuentas, sin interrumpir las operaciones existentes ni comprometer la funcionalidad actual.

El sistema legado está hecho está hecho en Java, con BD Oracle e implementa los siguientes módulos: Procesamiento de Pagos y Transferencias, Gestión de Préstamos y Créditos, Gestión de Productos Financieros, Contabilidad y Reportes Financieros, Cumplimiento y Seguridad, Gestión de Clientes (CRM), Integración y Servicios Externos, Gestión de Cuentas.

El módulo de gestión de cuentas administra las cuentas de los clientes, incluyendo la apertura, cierre, y mantenimiento de cuentas de ahorro y corrientes. Está conectado con módulos de:

1. **Procesamiento de Pagos y Transferencias:** Para asegurar que cualquier movimiento de fondos se registre y actualice el saldo de la cuenta en tiempo real, manteniendo la integridad y precisión de las operaciones.
2. **Gestión de Préstamos y Créditos:** Para automatizar la administración de préstamos, incluyendo el desembolso y la actualización del saldo de la cuenta después de cada transacción relacionada con el préstamo.
3. **CRM:** (Para gestionar la información y el historial de los clientes, lo que incluye detalles sobre las cuentas bancarias que poseen, sus preferencias, y su comportamiento financiero) para coordinar operaciones relacionadas con cuentas. El siguiente es el modelo de datos del sistema core bancario.

Veamos cómo aplicar el patrón strangler applicationhaciendo uso de Why driven design

Después de conversar con los ejecutivos de cangrecorp concluimos que la Gestión de cuentas es clave en el crecimiento de la empresa porque desde allí se administran todas las operaciones de transferencia y saldo. Por esta razón es necesario migrar el componente a una arquitectura que cumpla con la siguientes características:

- **Modularización y escalabilidad:** extraer el módulo de gestión de cuentas del sistema monolítico legado y convertirlo en un sistema independiente. El sistema debe ser escalable para manejar el crecimiento previsto en el número de usuarios y transacciones.

- **Flexibilidad y adaptabilidad:** el sistema debe permitir la adición de nuevas funcionalidades y la modificación de las existentes sin afectar otras partes del sistema. Debe ser posible integrar nuevas tecnologías y servicios.

- **Alta disponibilidad y resiliencia:** el microservicio debe ser altamente disponible, con un tiempo de actividad del 99.99%. Implementar mecanismos de recuperación ante fallos y balanceo de carga para asegurar que el servicio siga funcionando sin interrupciones.

- **Integración con el sistema legado:** mantener la interoperabilidad con los módulos restantes del sistema legado (como procesamiento de pagos, préstamos y crm). El sistema debe poder leer y escribir en la base de datos legada sin afectar la integridad de los datos ni la consistencia transaccional.

- **Seguridad y cumplimiento:** garantizar que el sistema cumpla con todas las regulaciones financieras y de seguridad, como las leyes de protección de datos en cada país donde opera.

- **Migración sin interrupción:** el proceso de extracción y migración debe ser gradual y sin causar interrupciones en el servicio.

2. Entender el contexto y los objetivos de negocio

Luego de tener algunas conversaciones de negocio identificamos algunos datos relevantes que guiarán el proceso de entendimiento del sistema legado y diseño de la arquitectura objetivo. Cangrecorp se dirige a clientes en Colombia, Estados Unidos y Puerto Rico. 3 millones en Colombia, 4 millones en Estados Unidos, y 0.5 millones en Puerto rico y espera crecer un 20% anual en cada uno de estos países en los próximos 5 años. El siguiente es su modelo de negocio real.

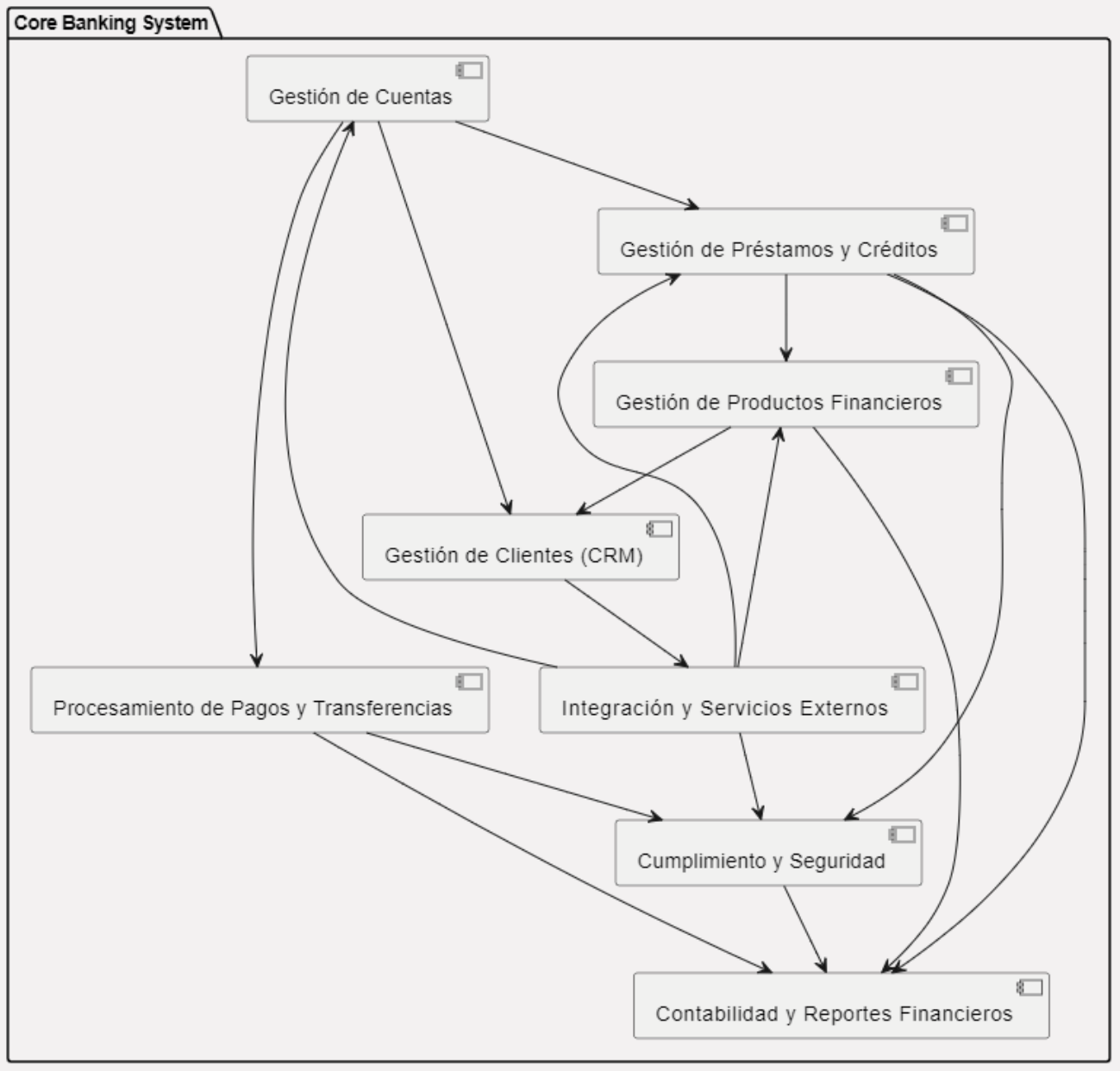
<table>
<tr>
<td rowspan="2">Asociados clave:
Inversionistas.
Product Owner
Product manager
Marcas(VISA, MC, EXPRESS)
Área Legal
Área de Operaciones
Área de Innovación
Servicio al Cliente
Soporte de página de web
Soporte de app móvil</td>
<td>Actividades clave:
Disponibilidad de acceder a la plataforma mediante su aplicación en PlayStore y AppStore.
Publicidad en RRSS
Marketing Publicitario.
Usuarios puedan recomendar apps y servicios
Digitalizar procesos que se puedan insertar en los medios digitales (página web y app móvil)</td>
<td rowspan="2">Oferta de valor:
Transacciones rápidas y no engorrosas.
Soluciones auto-eficientes.
Realizar pago de préstamos vía whatsapp.
Lenguaje coloquial y no técnico.
Cobro de pocas comisiones
Ofrecimiento de productos inclinados a millennials
Digitalización y funcionalidades innovadoras
Educación financiera con tips de ahorro y uso correcto de tarjetas de crédito
Vinculación de códigos de seguridad con mensajes de texto</td>
<td>Relación con clientes:
Redes sociales: instagram, facebook, twitter, mailing.
Mailing personalizado con ofrecimientos de productos
Mailing con tips cortos de uso eficiente de web y app
Soporte online por medio de chatbot o videochat
Ofrecimiento de ofertas acorde a movimientos de clientes
Mismo lenguaje en todos los canales
Simuladores de créditos de fácil acceso</td>
<td rowspan="2">Segmento de mercado:
Todo tipo de público.
Objetivo: millennials. (18-35)

Usuarios que:
Son tecnológicos
Interactúan constantemente en redes sociales.
Son críticos
Les gusta los procesos fácil y rápidos.
Quiere una experiencia 100% digitalizada, los procesos manuales son cosas del pasado.
Confían en personas, no en compañías.
Se guían de recomendaciones y opiniones de otros usuarios en redes sociales</td>
</tr>
<tr>
<td>Recursos clave:
Aplicaciones web.
Plataforma tecnológica.
Renovación de alianzas.
Servicios online 24 horas
Soporte online 24 horas</td>
<td>Canales:
Sitio web.
App móvil en PlayStore (android) y Apple (IOS).
Mail a clientes clásicos con invitación a la digitalización
Medios de transacciones aliados como: cajeros.
Agentes
Banca telefónica
Redes Sociales</td>
</tr>
<tr>
<td colspan="3">Estructura de costos:
Plataforma tecnológica.
Marketing.
Hallar inversionistas.
Transacciones realizadas por clientes</td>
<td colspan="2">Fuentes de ingreso:
Cobro de intereses con los créditos otorgados por diferentes canales y en sus diferentes modalidades</td>
</tr>
</table>

2. Entender el contexto y los objetivos de negocio

Tuvimos algunas conversaciones técnicas y nos apoyamos en inteligencia artificial generativa para entender los sistemas legados. En este punto sacamos muchos diagramas de arquitectura actual que nos permitieron identificar los módulos del sistema legado y sus dependencias para responder las siguientes preguntas:

- ¿Qué módulos de negocio implementa el sistema legado?
- ¿Cómo podríamos extraer los módulos de manera evolutiva manteniendo compatibilidad con el modelo de datos que usan los demás módulos?
- ¿Cómo hacer le evolución sin afectar a los sistemas externos y front ends que usan el sistema monolítico?



4. Diseñar de manera colaborativa y estructurada

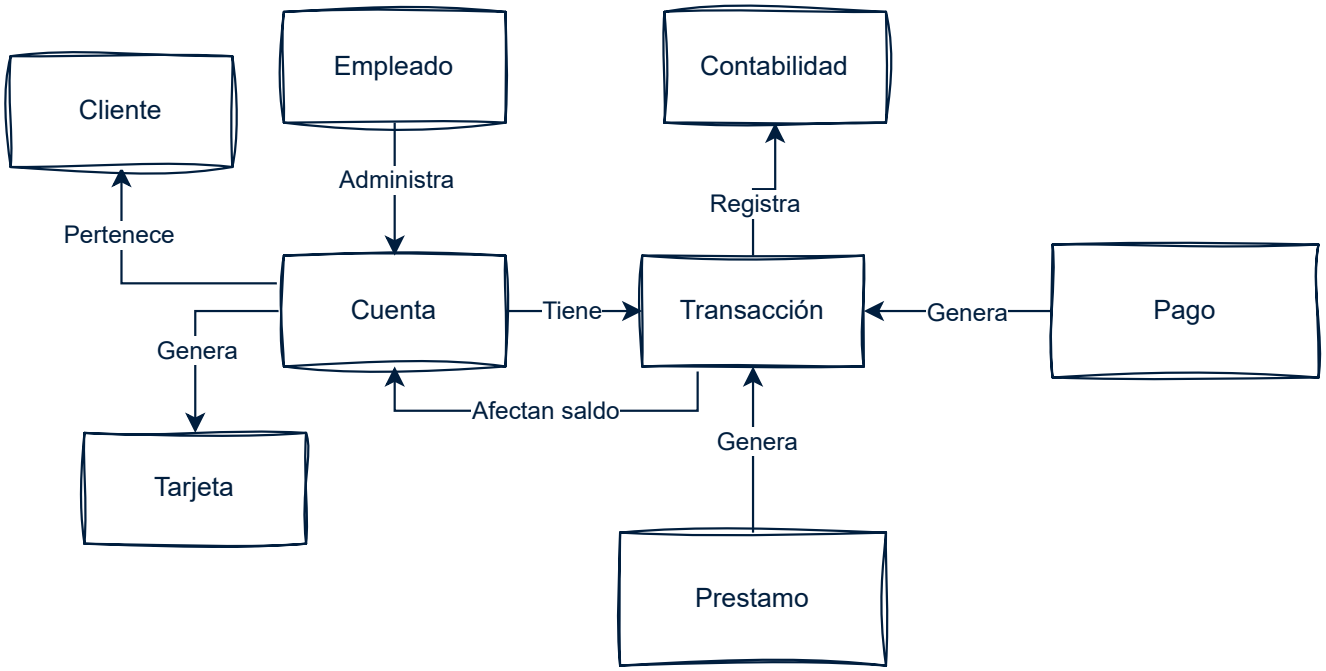
4.1 Identificar drivers de arquitectura

Realizamos algunos talleres para identificar los requisitos arquitectónicamentesignificativos, los atributos de calidad, escenarios de atributos de calidad,restricciones técnicas y de negocio.

4.2 Realizar el diseño estratégico

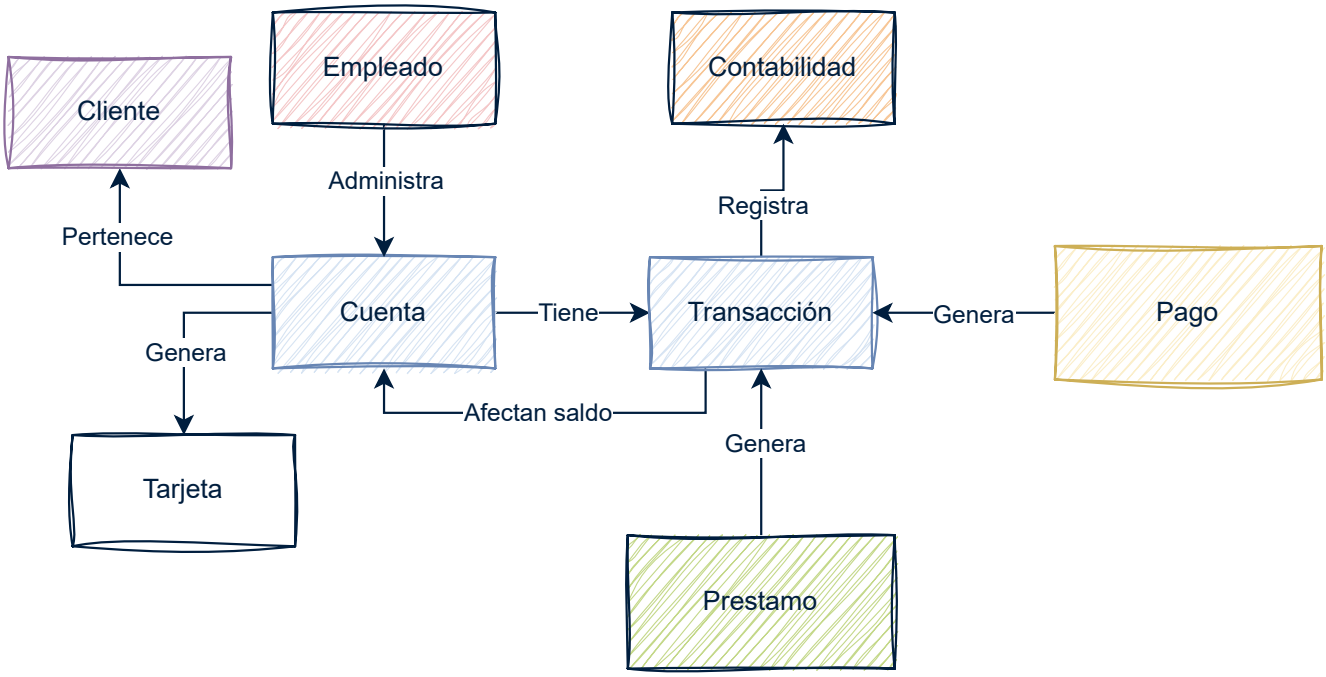
4.2.1 Descubrir el dominio

A partir del entendimiento de negocio y del sistema legado, identificamos elsiguiente modelo de dominio (Es una simplificación del modelo completo parafacilitar el entendimiento)

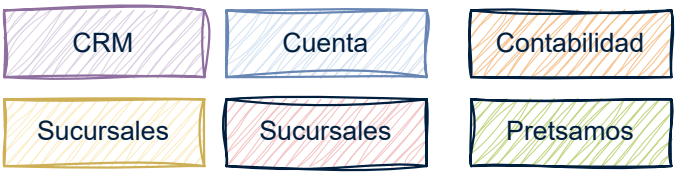


4.2.2 Descomponer y conectar el dominio

Luego analizamos la estructura de negocio y aplicamos heurísticas para identificar sub dominios y bounded contexts.



Dominios y sub dominios



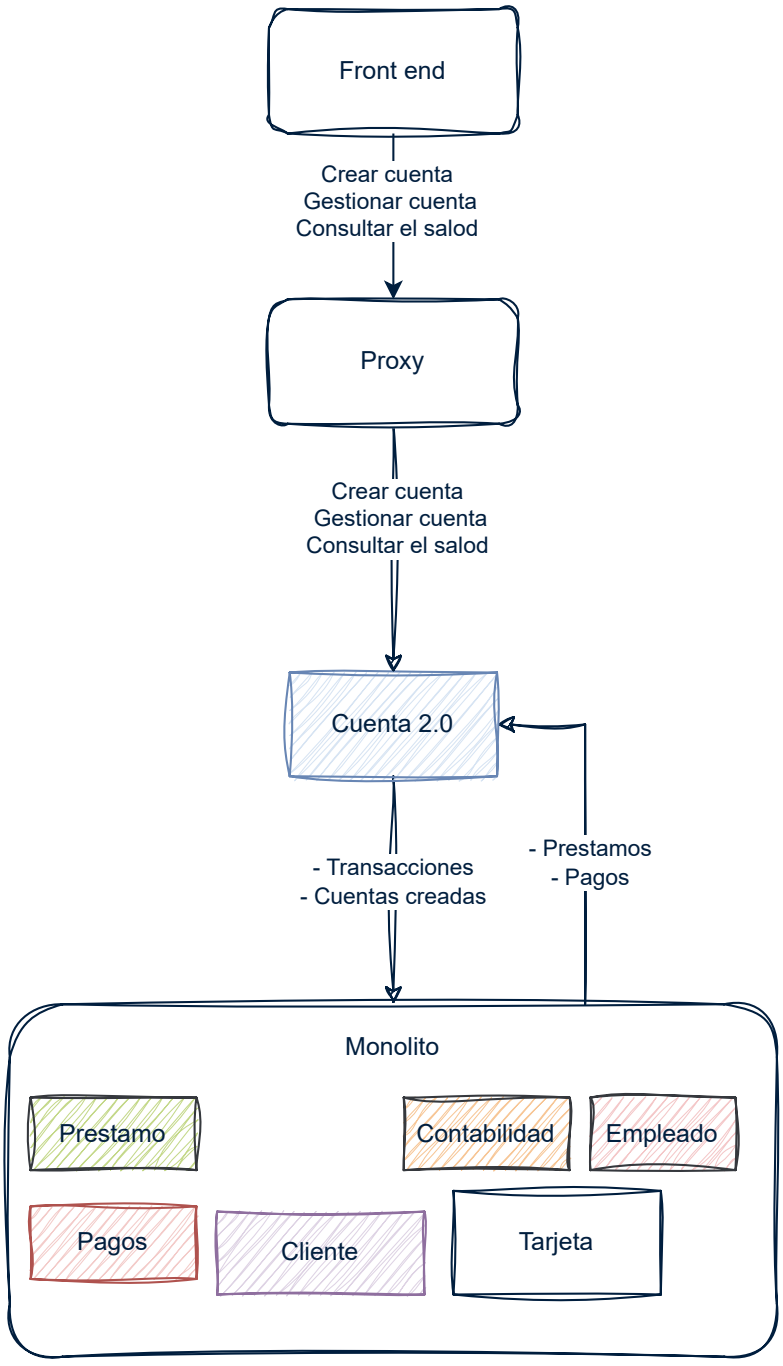
4.2.3 Clasificar el dominio

No es parte del alcance de este e-book profundizar en este paso, si quieres más detalles ve al E-Book Why Driven Design para diseño de arquitectura de soluciones ágil.

4.2.4 Identificar estrategias de migración y co existencia

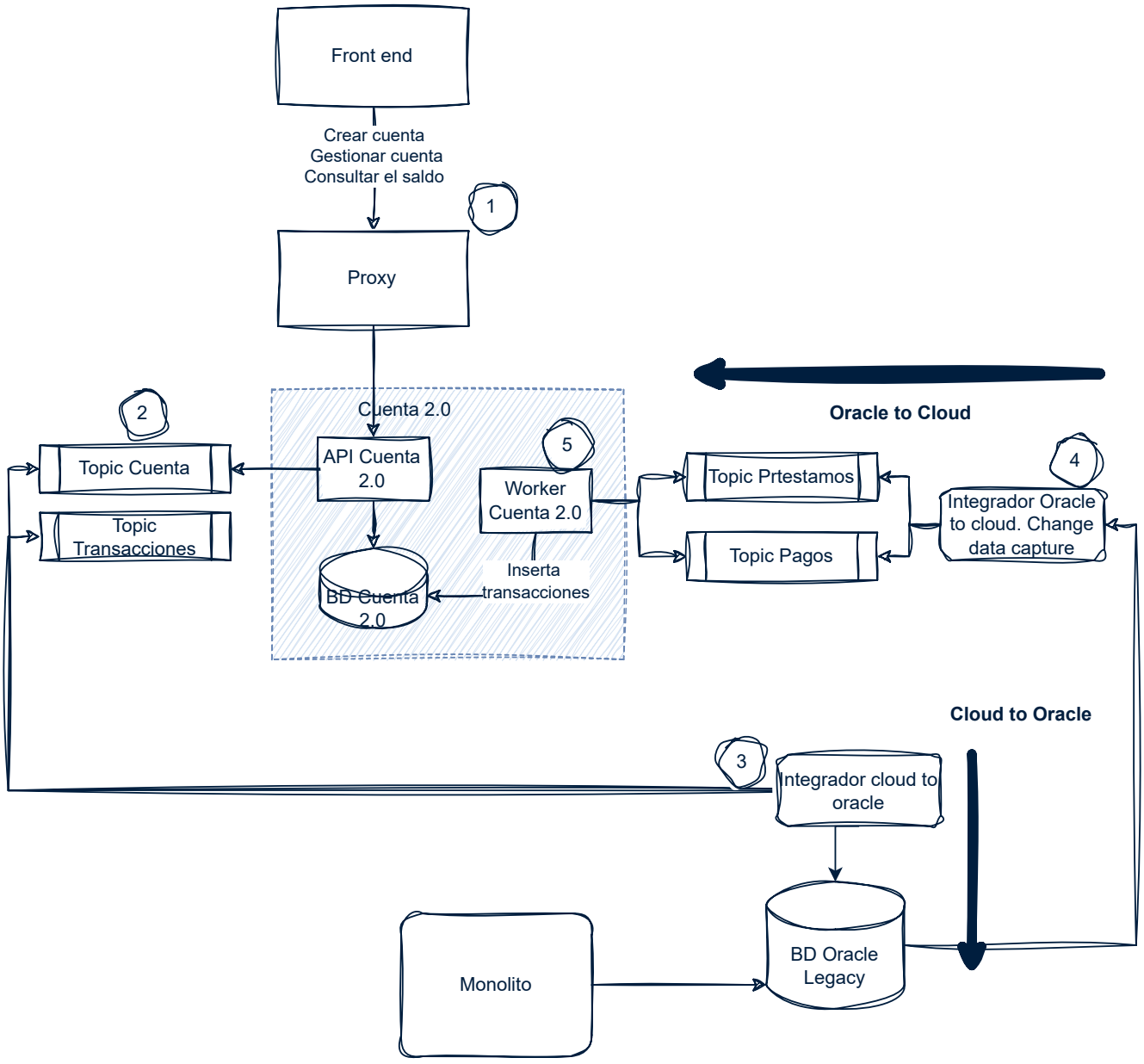
Ahora es momento de establecer la estrategia general para extraer el módulo decuentas identificando los flujos de sincronización de datos que deben implementarse.

La estrategia que definimos es que el front end se mantendría igual pero haría llamados al nuevo sistema de cuenta 2.0, quien a su vez mantendría compatibilidad con el legado sincronizando las transacciones y cuentas creadas ala BD monolítica y su vez recibiría los datos de prestados y pagos procesados en el legado para actualizar la información de transacciones.



4.3 Realizar el diseño técnico

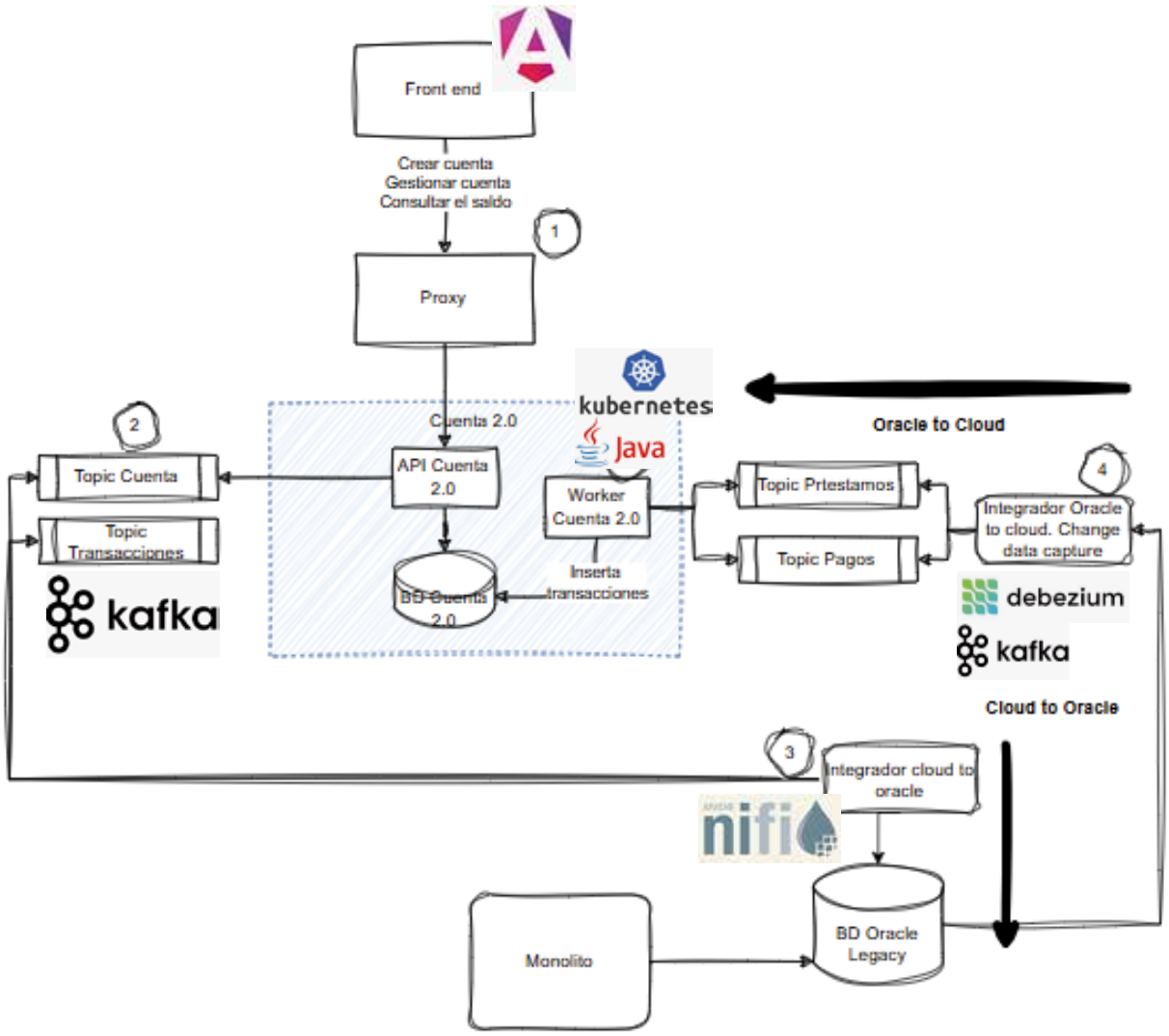
Finalmente debemos establecer las tecnologías y asignar responsabilidades a loscomponentes para dar vida al diseño estratégico.



1. El Api de cuenta 2.0 recibe todas las solicitudes realizadas desde los front end y aplicaciones cliente.
2. La información de las cuentas creadas y transacciones procesas es llevada a un sistema de mensajería para notificar a otras aplicaciones.
3. Un componente integrador Oracle to cloud se suscribe a los mensajes e inserta la información en la base de datos Oracle para mantener la compatibilidad con losmódulos legados.
4. La información de los prestamos y pagos es extraída en tiempo real por un integrador cloud to oracle que implementa tecnología change data capture.
5. Un worker procesador asincrónico que es parte del nuevo sistema Cuenta 2.0 sincroniza los datos y los almacena en su propia base de datos.

4.4 Realizar la instanciación tecnológica.

Seleccionar las tecnología es una labor relativamente sencilla una vez se tienen los componentes de la arquitectura técnica definidos. A continuación se describen lastecnologías a usar: Kafka, Angula, Java, Kubernetes, Nifi, Debezium.



Conclusiones

La migración evolutiva de sistemas legados hacia arquitecturas componibles es una estrategia que permite a las organizaciones adaptarse a los rápidos cambios tecnológicos y a las crecientes demandas del mercado. A lo largo del libro, exploramos el potencial de esta metodología para transformar sistemas obsoletos en soluciones flexibles, escalables y resilientes, sin interrumpir las operaciones.

La migración evolutiva permite realizar una transición gradual, lo cual es fundamental para minimizar riesgos. En lugar de una reescritura o reemplazo total, esta metodología apuesta por la descomposición progresiva del sistema legado en componentes modulares que puedan actualizarse, ampliarse o reemplazarse conforme sea necesario. Esto no solo reduce el impacto en el negocio, sino que también facilita el crecimiento ágil y adaptable de la infraestructura tecnológica, permitiendo a las empresas responder rápidamente a nuevas oportunidades ydesafíos.

Las arquitecturas componibles promueven el uso de módulos o componentes autónomos y reutilizables, que pueden ensamblarse para formar soluciones adaptadas a necesidades específicas.

El patrón Strangler Application es uno de los métodos más efectivos para migrar progresivamente funcionalidades críticas de sistemas legados hacia plataformas modernas. Al permitir que los sistemas nuevos crezcan en paralelo con los sistemas antiguos, este patrón minimiza los riesgos asociados con la migración y aseguraque el negocio pueda seguir operando sin interrupciones significativas.

La migración de sistemas requiere una estrategia robusta de integración ysincronización de datos entre los sistemas legados y las nuevas aplicaciones. La implementación de herramientas ETL (Extracción, Transformación y Carga) de bajo código, change data capture y el uso de APIs de integración en tiempo real son componentes esenciales en este proceso. Estas herramientas permiten una sincronización continua, lo que garantiza que ambos sistemas mantengan la coherencia de datos durante la transición y reduce la complejidad en la migraciónde funcionalidades críticas.

Para aquellos interesados en profundizar y aplicar estos conocimientos, losinvitamos a inscribirse en nuestro curso especializado:

Diseño evolutivo y modernización de aplicaciones legadas.

Este curso ofrece un recorrido detallado ypráctico por los conceptos, estrategias y herramientas necesarias para llevaradelante proyectos de modernización de sistemas con éxito. Al final del curso,tendrás la habilidad de diseñar, implementar y gestionar un proceso evolutivo que permita a tu organización adaptarse a los desafíos tecnológicos actuales y futuros.

Concluimos con una invitación a ver la modernización no solo como una tarea técnica, sino como una oportunidad para construir una base tecnológica que pueda crecer y transformarse junto con la organización. El camino hacia la modernización de sistemas legados puede ser desafiante, pero al adoptar una metodología evolutiva, las organizaciones pueden crear un entorno tecnológico que no solo respalde, sino que impulse la innovación y el éxito a largo plazo.

En nuestros 20 años de experiencia haciendo arquitectura, continuamos evidenciando la importancia de esta práctica como un habilitador para generar conexión entre negocio y tecnología. El diseño evolutivo de sistemas legados es una práctica esencial para mantener la competitividad empresarial.

Si quieres conocer más te invitamos a tomar el **curso gratuito de fundamentos de arquitectura de soluciones ágil.**

Escuela de Arquitectura Tecnológica.